Active Learning for Classifying Political Tweets

Tjong Kim Sang, Erik; Esteve Del Valle, Marc; Kruitbosch, Herbert; Broersma, Marcel

Published in:
International Science and General Applications

Active Learning for Classifying Political Tweets

Erik Tjong Kim Sang[1], Marc Esteve del Valle[2],
Herbert Kruitbosch[2], and Marcel Broersma[2]

[1]Netherlands eScience Center - Amsterdam - The Netherlands -
e.tjongkimsang@esciencecenter.nl
[2]University of Groningen - The Netherlands -
{m.esteve.del.valle,h.t.kruitbosch,m.j.broersma}@rug.nl

Abstract

We examine methods for improving models for automatically labeling social media data. In particular we evaluate active learning: a method for selecting candidate training data whose labeling the classification model would benefit most of. We show that this approach requires careful experiment design, when it is combined with language modeling.

1 Introduction

Social media, and in particular Twitter, are important platforms for politicians to communicate with media and citizens [1]. In order to study the behavior of politicians on Twitter, we have labeled tens of thousands political tweets

written in four languages (Dutch, English, Swedish and Italian) with respect to several categories, like function and topic. Labeling tweets is a time-consuming manual process which requires training of the human annotators. We would like to minimize the effort put in labeling future data and therefore we are looking for automatic methods for classifying tweets based on our annotated data sets.

The task of automatically assigning class labels to tweets is a variant of document classification. This is a well-known task for which several algorithmic solutions are known [2]. A recently developed tool for document classification is fastText [3]. It consists of a linear classifier trained on bags of character n-grams. This is a useful feature for our task: in a compounding language like Dutch, useful information can be present at the character n-gram level. For example, if a word like *bittersweet* appears in the data only once, an n-gram-sensitive system could still pickup similarities between this word and the words *bitter* and *sweet*. FastText also includes learning language models from unlabeled text [4], an excellent feature for our task, where labeled data is scarce and unlabeled data is abundant.

In a typical time line of our work, we would study the tweets of politicians in the weeks preceding an election and then again in the weeks preceding the next election, some years later. Given the long time between the periods of interest, we expect that the classification model will benefit from having manually labeled data of each period. However, we would like to limit the human labeling effort because of constraints on time and resources. We will apply active learning [5] for selecting the best of the new tweets for the classification model, and label only a small selection of these tweets. Active learning has previously been used for reducing the size of candidate training data with more than 99%, without any performance loss [6].

The contribution of this paper is two-fold. Firstly, we will show that fastText

can predict a non-trivial class of our political data with reasonable accuracy. Secondly, we will outline how active learning can be used together with fastText. We found that this required careful experiment design.

After this introduction, we will present some related work in Section 2. Section 3 describes our data and the machine learning methods applied in this study. The results of the experiments are presented in Section 4. In Section 5, we conclude.

2 Related work

Social media have amplified the trend towards personalization in political communication. Attention has shifted from political parties and their ideological stances to party leaders and individual politicians [7]. One way of studying personalization, is by examining the behavior of politicians on social media, in particular during campaigns leading to an election. Studies have focused on various social media like Twitter [1], Facebook [8] and Instagram [9]. Because of its open nature, Twitter is especially popular for studying online political communication [10].

Document classification is a well-known task which originates from library science. Automatic methods for performing this task, have been available for more than twenty years, for example for spam filtering [11] and topic detection in USENET newsgroups [12]. While the restricted length of social media text poses a challenge to automatic classification methods, there are still several studies that deal with this medium [13, 14]. Popular techniques for automatic document classification are Naive Bayes [15] and Support Vector Machines [16]. Despite its relatively young age, fastText [3] has also become a frequently used tool for automatic document classification and topic modeling [17, 18]. The word vector-based language models used by fastText, were originally proposed

by Mikolov et al. [19].

The term of active learning was introduced in the context of machine learning in 1994 [20], referring to a form of learning where the machine can actively select its training data. Since then active learning has been applied in many contexts [5]. A well-known application in natural language processing was the study by Banko and Brill [6], which showed that with active learning, more than 99% of the candidate training data could be discarded without any performance loss.

In the study described in this paper, we employ labeled tweets developed by the Centre for Media and Journalism Studies of the University of Groningen [21]. Broersma, Graham et al. have performed several studies based on these data sets [22, 1, 23]. Most importantly for this paper, Tjong Kim Sang et al. [24] applied fastText to the Dutch 2012 part of the data set. They also evaluated active learning but observed only decreasing performance effects.

3 Data and methods

Our data consist of tweets from Dutch politicians written in the two weeks leading up to the parliament elections in The Netherlands of 12 September 2012. The tweets have been annotated by the Groningen Centre for Media and Journalism Studies [21]. Human annotators assigned nine classes to the tweets, among which tweet topic and tweet function. In this paper we exclusively deal with the tweet function class. This class contains information about the goal of a tweet, for example campaign promotion, mobilization, spreading news or sharing personal events. A complete overview of the class labels can be found in Table 3. A tweet can only be linked to a single class label.

The data annotation process is described in Graham et al. [1]. The tweets were processed by six human annotators. Each tweet was annotated by only one annotator, except for a small set of 300 randomly chosen tweets. The small tweet

subset was used for computing inter-annotator agreement for four classes with average pairwise Cohen kappa scores [25]. The kappa scores were in the range 0.66–0.97. The function class proved to be the hardest to agree on: its kappa score was 0.66. This corresponds with an pairwise inter-annotator agreement of 71%.

Twitter assigns a unique number to each tweet: the tweet id. We found that the data set contained some duplicate tweet ids. We removed all duplicates from the data set. This left 55,029 tweets. They were tokenized with the Python's NLTK toolkit [26] and converted to lower case. Next we removed tokens which we deemed useless for our classification model over long time frames: reference to other Twitter users (also known as tweet handles), email addresses and web addresses. These were replaced by the tokens USER, MAIL and HTTP. Finally the tweets were sorted by time and divided in three parts: test (oldest 10%), development (next 10%) and train (most recent 80%). We chose to have test and development data from one end of the data set because there are strong time dependencies in the data. Random test data selection would have increased the test data scores and would have made the scores less comparable with the scores that could be attained on other data sets.

We selected the machine learning system fastText [3] for our study because it is easy to use, performs well and allows for incorporation of language models. We only changed one of the default parameter settings of fastText: the size of the numeric vectors used for representing words in the text (dim): from 100 to 300. The reason for this change was that pretrained language models often use this dimension, for example models derived from Wikipedia [4]. By using the same dimension, it becomes easier to use such external language models and compare them with our own[1]. We explicitly set the minimal number of word

[1]See Tjong Kim Sang et al. [24] for a comparison between models build from tweets and models build from Wikipedia articles.

occurrences to be included in the model (minCount) to 5. This should be the default value for this parameter but we have observed that fastText behaves differently if the parameter value is not set explicitly.

Because of the random initialization of weights in fastText, experiment results may vary. In order to be able to report reliable results, we have repeated each of our experiments at least ten times. We will present average scores of these repeated results. We found that the test evaluation of fastText (version May 2017) was unreliable, possibly because some test data items are skipped during evaluation. For this reason we did not use the test mode of the tool but rather made it predict class labels which were then compared to the gold standard by external software [27].

In active learning, different strategies can be used for selecting candidate training data. In this study, we compare four informed strategies with three baselines. Three of the informed strategies are variants of uncertainty sampling [5]. The machine learner labeled the unlabeled tweets and the probabilities it assigned to the labels were used to determine the choices in uncertainty sampling. As an alternative, we have also experimented with query-by-committee [5]. We found that its performance for our data was similar to uncertainty sampling.

The data selection strategies used in this study are:

Sequential (baseline) choose candidate training data in chronological order, starting with the oldest data. Because there are strong time-dependent relations in our data, we also evaluate the variant **Reversed sequential** (baseline) which selects the most recent data first.

Random selection (baseline) randomly select data.

Longest text choose the longest data items first, based on the number of characters.

Least confident first select the data items with an automatically assigned label with the lowest probability.

Margin choose the data with of which the probability of the second-most likely label is closest to the probability of the most likely label

Entropy first select data items of which the entropy of the automatic candidate labels is highest.

The methods Entropy, Margin and Least confident select the data the machine learner is least confident of while Longest text selects the data that are most informative. The entropy is computed with the standard formula $-\sum_i p_i * log_2(p_i)$ [28] where p_i is the probability assigned by the machine learner to a candidate training data item in association with one of the twelve class labels.

In their landmark paper, Banko and Brill [6] observed that having active learning select all the new training data, resulted in the new data being biased toward difficult instances. They solved this by having active learning select only half of the new training data, while selecting the other half randomly. We will adopt the same approach. Dasgupta [29] provides another motivation for this strategy: the bias of an initial model might prevent active learning from looking for solutions in certain parts of the data space. Incorporating randomly chosen training items can help the model to overcome the effect of this bias.

4 Experiments and analysis

We started our study with an assessment of active learning and our software. For this purpose we attempted to reproduce part of the study by Banko and Brill [6] on disambiguation of confusable words. We focused on one specific set of confusable words: *to*, *too* and *two*. The billion word text corpus of the study by

Banko and Brill [6] is not publically available. Instead, we used the billion word corpus developed Chelba et al [30]. Our version of this corpus is the same as in that paper: a total of 776,436,550 tokens (after removing duplicate sentences and not counting the sentence boundary tokens <s> and </s>) of tokenized sentences in a random order, split in a test part of 1% (7,790,025 tokens) and a train part of 99% (768,646,525 tokens).

From the data, we extracted all occurrences of the tokens *to*, *too* and *two*, with a context of five preceding tokens and five following tokens. In case one of the focus tokens occurred near the beginning or the end of a sentence, we added extra filler tokens to the extracted text snippets to make sure that all of them had the same length. Next, we removed the target focus token from the text snippets and trained fastText to predict them based on the rest of the snippet. As initial training data we used the first 0.1% of the full training data set (19,681 text snippets). This data set was subsequently increased with ten blocks of 0.1% of the data, half of which was selected by active learning and half of which was selected randomly, like described in section 3. We evaluated the active learning algorithms Margin, Entropy and Least confident and compared their performance with Random selection. Because the sentence order in our data set is randomized, random data selection was performed by selecting the first part of the available data. In order to restrict the experiment to a reasonable time, we restricted the part of the data available for active learning to the first 5% of the training data set.

The results of this experiment can be found in Figure 1. Like Banko and Brill [6], we found that active learning does indeed outperform random data selection for this data set. Margin (95.6%), Entropy (95.6%) and Least confident (95.6%) all outperformed Random selection (95.2%) after processing 1.1% of the training data. Margin even tied with the performance of Random selection

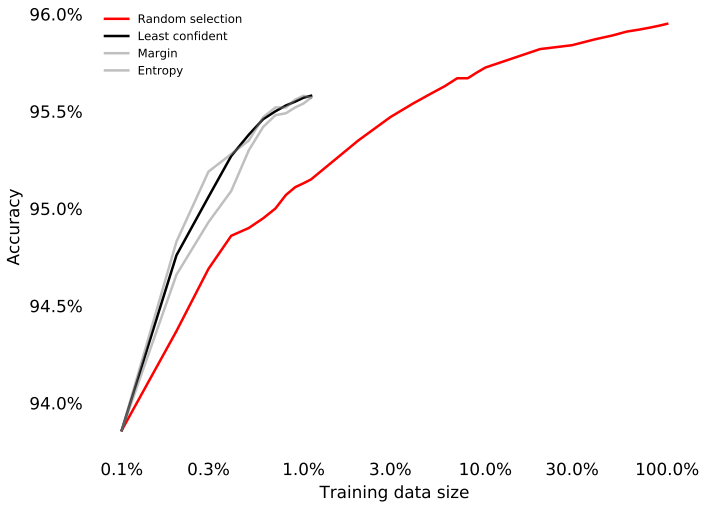

Figure 1: Rerun of an experiment of Banko and Brill [6]: disambiguation of the confusable words *to*, *too* and *two*, based on a context of five preceding and five following tokens, while training and testing on the billion word corpus of Chelba et al. [30]. The active learning algorithms Margin (black line), Entropy and Least confident all outperform Random selection (red line) after processing 1.1% of the training data.

after processing all data that was available to active learning. We performed one experiment with Margin active learning where the active learning data part was increased to 10% but that did not lead to an improved performance.

Next we attempted to reproduce the results reported by previous work on our main data set. Tjong Kim Sang [24] reported a baseline accuracy of 51.7±0.2% when training fastText on the most recent 90% of the data and testing on the oldest 10% (averaged over 25 runs). We repeated this experiment and derived a model from the train and development parts of the data set and evaluated this model on the test part. We obtained an accuracy of 51.6±0.7%, averaged over 10 runs, which is similar to the earlier reported score. This baseline score is not very high but as the low pairwise interannotator agreement (71%) showed, this

is a difficult task.

As an additional check of how well fastText performed on our data set, we compared it with the performance of a state-of-the-art machine learning technique: deep learning [31]. This machine learning method is available in many different variants. We chose the multilayer perceptron, a neural network with several hierarchically organized layers [32] connected by weights which are updated by backpropagation [33]. We reused most of the configuration and the parameter setting of the Reuters example of the API Keras[2], like five layers, five epochs and batch size 32, although we increased the maximum vocabulary size to 10,000. The multilayer perceptron achieved an accuracy of 50.3% on our data set, significantly worse than fastText. Since the training phase of fastText was also a lot shorter than that of the multilayer perceptron, we concluded that fastText was an excellent choice for our task.

In this study, we will compare several techniques and select the best. In order to avoid overfitting, we will leave this data set alone. Unless mentioned otherwise, scores reported in this paper will have been derived from testing on the development data part after training on the training data, or a part of the training data. We repeat the initial experiment, this time training fastText on the train data and evaluating on the development data. We obtained an average accuracy over ten runs of 54.2±0.4%, which shows that the labels of the development data are easier to predict than those of the test data.

Next, we evaluated active learning. Earlier, Tjong Kim Sang et al. [24] performed two active learning experiments. Both resulted in a decrease of performance when the newly annotated tweets were added to the training data. We do not believe that data quantity is the cause of this problem: their extra 1,000 tweets (2%) of the original training data size should be enough to boost performance (see for example Banko and Brill [6]'s excellent results with 0.7% of the

[2]The documentation of Keras can be found on https://keras.io

training data). However, the quality of the data could be a problem. The data from the active data set and the original data set were annotated by different annotators several years apart. While there was an annotation guideline [21], it is possible that the annotators interpreted it differently. It would have been better if both training data and the active learning data had been annotated by the same annotators in the same time frame.

In order to make sure that our data was consistently annotated, we only use the available labeled data sets. We pretend that the training data is unannotated and only use the available class labels for tweets that are selected by the active learning process. The process was split in ten successive steps. It started with an initial data set of 1.0% of all labeled data, selected with the Sequential strategy. FastText learned a classification model from this set and next 0.2% of the data was selected as additional training data: 0.1% with active learning and 0.1% randomly, as described in Section 3. These steps were repeated ten times. The final training data set contained 3.0% of all labeled data. In order to obtain reliable results, the active learning process was repeated 30 times.

The random initialization of fastText pose a challenge to a successful combination with active learning. During the training process, fastText creates numeric vectors which represent the words in the data. However, when we expand the training data set and retrain the learner on the new set, these word vectors might change. This could invalidate the data selection process: the newly selected training data might work fine with the old word vectors but not with the new word vectors. In order to avoid this problem, we need to use the same word vectors during an entire active learning experiment. This means that the word vectors needed to be derived for all of the current and future training data before each experiment, without using the data labels. We used the skipgram model for this, with the fastText parameter setting described in

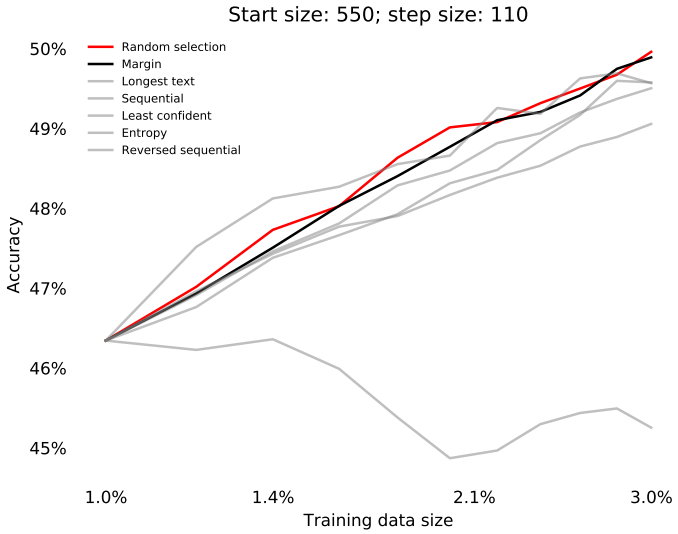

Figure 2: Performance of seven data selection methods, averaged over thirty runs. The Random selection baseline (red line) outperforms all active learning methods at 3.0% of the training data. Margin sampling (black line) is second best. There is no significant different between the accuracies of the best six methods at 3.0% (see Table 1). Note that the horizontal axis is logarithmic.

Section 3. A set of such word vectors is called a language model. Providing the machine learner with word vectors from these language models improved the accuracy score: from 54.2±0.4% to 55.6±0.3%.

The results of the active learning experiments can be found in Figure 2 and Table 1. All the data selection strategies improve performance with extra data, except for the Reversed sequential method. The initial 1.0% of training data selected with the Sequential method was a good model of the development set, since it originated from the same time frame as the development data. The data from the Reversed sequential process came from the other end of the data set and was clearly less similar to the development set.

The differences between the other six evaluated methods proved to be insignificant (see Table 1). It is unclear why neither Margin, nor Entropy, nor

Train size	Accuracy	Method
80.0%	55.6±0.3%	Ceiling (all training data)
3.0%	50.0±0.9%	Random selection
3.0%	49.9±0.9%	Margin
3.0%	49.6±0.7%	Longest text
3.0%	49.6±0.9%	Sequential
3.0%	49.5±1.0%	Least confident
3.0%	49.1±0.8%	Entropy
3.0%	45.3±1.3%	Reversed sequential
1.0%	46.3±0.8%	Baseline

Table 1: Results of active learning experiments after training on 3.0% of the available labeled data in comparison with training on 80.0%. The Random selection baseline outperforms all evaluated active learning methods on this data set, although most of the measured differences are insignificant. Margin sampling is second-best. Numbers after the scores indicate estimated error margins ($p < 0.05$).

Least confident could outperform the Random selection baseline. Perhaps the method for estimating label probabilities (fastText-assigned confidence scores) was inadequate. However, we also evaluated bagging for estimating label probabilities and this resulted in similar performances. The Longest text method did not have access to as much information as the other three informed methods. It would be interesting to test a smarter version of this method, for instance one that preferred words unseen in the training data.

It is tempting to presume that if Margin, Longest text, Least confident and Entropy perform worse than Random selection, then their reversed versions must do better than this baseline. We have tested this and found that this was not the case. Shortest text (49.1%), Smallest entropy (48.9%), Largest margin (48.9%) and Most confident (48.8%) all perform worse than Random selection and also worse than their original variant.

Since no active learning method outperformed the random baseline, we used Random selection for our final evaluation: selecting the best additional training data while evaluating on the data sets of Tjong Kim Sang at al [24]: train

Method	Train size	Accuracy
Baseline	90.0%	51.6±0.7%
+ language model	90.0%	55.5±0.4%
+ active learning data 1	90.2%	55.4±0.4%
+ active learning data 2	90.4%	55.6±0.5%
+ active learning data 3	90.6%	55.6±0.3%

Table 2: Results of active learning (with Random selection) applied to the test set. Additional pretrained word vectors improve the classification model but active learning does not.

(49,526 tweets), test (5,503) and unlabeled (251,279). A single human annotator labeled the selected tweets. At each iteration 110 tweets were selected randomly. After labeling, the tweets were added to the training data and the process was repeated. Three iterations were performed. Each of them used the same set of skipgram word vectors, obtained from all 300,805 non-test tweets.

The result of this experiment can be found in Table 2. The extra training data only marginally improved the performance of the classifier: from 55.5% to 55.6%. The improvement was not significant. This is surprising since we work with the same amount of additional data as reported in Banko and Brill [6]: 0.6%. They report an error reduction of more than 50%, while we find no effect.

However, the percentages of added data do not tell a complete story. A close inspection of Figure 4 of the Bank and Brill paper shows that the authors added 0.6% of training data to 0.1% of of initial training data. This amounts to increasing the initial training data with 600%, which must have an effect on performance, regardless of the method used for selecting the new data. Instead, we add 0.6% to 90% of initial training data, an increase of only 0.7%. Unfortunately, we don't have the resources for increasing the data volume by a factor of seven. The goal of our study was to improve classifier performance with a small amount of additional training data, not with a massive amount of extra data.

If relative data volumes are not enough to explain the differences between

Class	**Frequency**		**Frequency**	
Campaign Promotion	12,017	(22%)	53	(16%)
Campaign Trail	10,681	(19%)	61	(18%)
Own / Party Stance	9,240	(17%)	50	(15%)
Critique	8,575	(16%)	71	(21%)
Acknowledgement	6,639	(12%)	32	(10%)
Personal	4,208	(8%)	19	(6%)
News/Report	1,662	(3%)	32	(10%)
Advice/Helping	1,292	(2%)	0	(0%)
Requesting Input	307	(1%)	0	(0%)
Campaign Action	216	(0%)	0	(0%)
Other	116	(0%)	12	(4%)
Call to Vote	76	(0%)	0	(0%)
All data	55,029	(100%)	330	(100%)

Table 3: Distribution of the function labels in the annotated data set of 55,029 Dutch political tweets from the parliament elections of 2012 (left) and the 330 tweets selected with active learning (right). The 2012 data contain more campaign-related tweets while the active learning data contain more critical, news-related and non-political tweets (class Other).

Table 1 and Table 2, there could be two other causes. First, the distribution of the labels of the active learning data is different from that of the original data. The latter were collected in the two weeks before the 2012 Dutch parliament elections while the first were from a larger time frame: 2009-2017. We found that the original data contained more campaign-related tweets, while the active learning data had more critical, news-related and non-political tweets (Table 3).

The second reason for the differences between Tables 1 and 2 could be low inter-annotator agreement. We have included 110 tweets from the training data in each iteration, to enable a comparison of the new annotator with the ones from 2012. While Graham et al. [1] reported an inter-annotator agreement of 71% for the 2012 labels, we found that the agreement was of the new annotator with the previous ones was only 65%, despite the fact that the annotator had access to the guesses of the prediction system. A challenge for the annotator was that some of the contexts of tweets that earlier annotators had access to, was not available on Twitter anymore and therefore could not be used for choosing

the most appropriate label. The resulting lower quality of the new labels might have prevented the machine learner from achieving better performances.

5 Concluding remarks

We have evaluated a linear classifier in combination with language models and active learning on predicting the function of Dutch political tweets. In the process, we have improved the best accuracy achieved for our data set, from 54.8% [24] to 55.6%. We found that combining the classifier fastText with active learning was not trivial and required careful experiment design, with pretrained word vectors, parameter adjustments and external evaluation procedures. In a development setting, none of the evaluated four informed active learning performed better than the random baseline, although the performance differences were insignificant. In a test setting with the best data selection method (random sampling), we measured no performance improvement. The causes for this could be the small volume of the added data, label distribution differences between the new and the original training data and the fact that it was hard for annotators to label the data consistently.

We remain interested in improving the classifier so that we can base future data analysis on accurate machined-derived labels. One way to achieve this, would be re-examine the set of function labels chosen for our data set. We are currently evaluating the effect of collapsing labels. When we combine labels in such a way that we have six rather than twelve different labels, the classifier is able to predict the labels with an accuracy close to 70%, which is the minimum accuracy we require for follow-up work. However, we still have to determine if after the label collapse the labels still are interesting enough for further analysis.

An alternative to collapsing labels is splitting labels, for example by creating a separate binary label for each current label value. This would make possible

assigning multiple labels to one tweet, freeing the current burden of annotators of having to choose a single label even in cases where three or four different labels might be plausible. Making the task of the annotators easier would improve the inter-annotator agreement and may even improve the success of applying active learning to this data set.

How to best split the labels while still being able to use the current labels in the data, remains a topic for future work.

6 Acknowledgments

The study described in this paper was made possible by a grant received from the Netherlands eScience Center. We would like to thank three anonymous reviewers for valuable feedback on an earlier version of this paper.

References

[1] Todd Graham, Dan Jackson, and Marcel Broersma. New platform, old habits? Candidates use of Twitter during the 2010 British and Dutch general election campaigns. *New Media & Society*, 18:765–783, 2016.

[2] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys (CSUR)*, 34, 2002.

[3] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 427–431. ACL, Valencia, Spain, 2017.

[4] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association of Computational Linguistics*, 5:135–146, 2017.

[5] Burr Settles. *Active Learning Literature Survey.* Computer Sciences Technical Report 1648, University of Wisconsin-Madison, 2010.

[6] Michele Banko and Eric Brill. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th annual meeting on Association for Computational Linguistics*, pages 26–33. Association for Computational Linguistics, 2001.

[7] Peter Van Aelst, Tamir Sheafer, and James Stanyer. The personalization of mediated political communication: A review of concepts, operationalizations and key findings. *Journalism*, 13:203–220, 2012.

[8] Gunn Sara Enli and Eli Skogerbø. Personalized campaigns in party-centered politics. *Information, Communication & Society*, 16(5):757–774, 2013.

[9] Younbo Jung, Ashley Tay, Terence Hong, Judith Ho, and Yan Hui Goh. Politicians strategic impression management on instagram. In *Proceedings of the 50th Hawaii International Conference on System Sciences (HICSS)*. IEEE, Waikoloa Village, HI, USA, 2017.

[10] Andreas Jungherr. Twitter use in election campaigns: A systematic literature review. *Journal of Information Technology & Politics*, 13(1):72–91, 2016.

[11] Ion Androutsopoulos, John Koutsias, Konstantinos V. Chandrinos, George Paliouras, and Constantine D. Spyropoulos. An evaluation of naive bayesian anti-spam filtering. In G. Potamias, V. Moustakis, and M. van

Someren, editors, *Proceedings of the workshop on Machine Learning in the New Information Age*, pages 9–17. Barcelona, , Spain, 2000.

[12] Scott A. Weiss, Simon Kasif, and Eric Brill. *Text Classification in USENET Newsgroups: A Progress Report.* AAAI Technical Report SS-96-05, 1996.

[13] Liangjie Hong and Brian D. Davidson. Empirical study of topic modeling in Twitter. In *Proceedings of the First Workshop on Social Media Analytics (SOMA'10).* ACM, Washington DC, USA, 2010.

[14] Wayne Xin Zhao, Jing Jiang, Jianshu Weng, Jing He, Ee-Peng Lim, Hongfei Yan, and Xiaoming Li. Comparing Twitter and Traditional Media Using Topic Models. In *ECIR 2011: Advances in Information Retrieval*, pages 338–349. Springer, LNCS 6611, 2011.

[15] Andrew McCallum and Kamal Nigam. A Comparison of Event Models for Naive Bayes Text Classification. In *AAAI-98 workshop on learning for text categorization*, pages 41–48, 1998.

[16] Larry M. Manevitz and Malik Yousef. One-Class SVMs for Document Classification. *Journal of Machine Learning Research*, 2:139–154, 2001.

[17] Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. Deep Learning for Hate Speech Detection in Tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion.* International World Wide Web Conferences Steering Committee, 2017.

[18] Francesco Barbieri. Shared Task on Stance and Gender Detection in Tweets on Catalan Independence - LaSTUS System Description. In *Proceedings of the Second Workshop on Evaluation of Human Language Technologies for Iberian Languages (IberEval 2017).* Murcia, Spain, 2017.

[19] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv*, 1301.3781, 2013.

[20] David Cohn, Les Atlas, and Richard Ladner. Improving generalization with active learning. *Machine Learning*, 15:201–221, 1994.

[21] The Groningen Center for Journalism and Media Studies. *The Tweeting Candidate: The 2020 Dutch General Election Campaign: Content Analysis Manual*. University of Groningen, 2013.

[22] Todd Graham, Marcel Broersma, Karin Hazelhoff, and Guido van t Haar. Between broadcasting political messages and interacting with voters: The use of twitter during the 2010 uk general election campaign. *Information, Communication and Society*, 16:692–716, 2013.

[23] Marcel Broersma and Marc Esteve Del Valle. Automated analysis of online behavior on social media. In *Proceedings of the European Data and Computational Journalism Conference*. University College Dublin, 2017.

[24] Erik Tjong Kim Sang, Herbert Kruitbosch, Marcel Broersma, and Marc Esteve del Valle. Determining the function of political tweets. In *Proceedings of the 13th IEEE International Conference on eScience (eScience 2017)*, pages 438–439. IEEE, Auckland, New Zealand, 2017.

[25] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1), 1960.

[26] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc., 2009.

[27] Erik Tjong Kim Sang. *Machine Learning in project Online Behaviour*. Software repository available at https:// github.com/online-behaviour/machine-learning, 2017.

[28] C.E. Shannon. A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27(3), 1948.

[29] Sanjoy Dasgupta. Two faces of active learning. *Theoretical Computer Science*, 412:1756–1766, 2011.

[30] Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. One billion word benchmark for measuring progress in statistical language modeling. Technical report, Google, 2013.

[31] François Chollet. *Deep Learning with Python*. Manning Publications Co., 2018.

[32] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.

[33] David E. Rumelhart and James L. McClelland. *Parallel Distributed Processing*. The MIT Press, 1986.